



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

DAKTOOLS

P. Tipton, S. Brandon

August 9, 2004

Dakota Users' Group
Livermore, CA, United States
August 2, 2004 through August 2, 2004

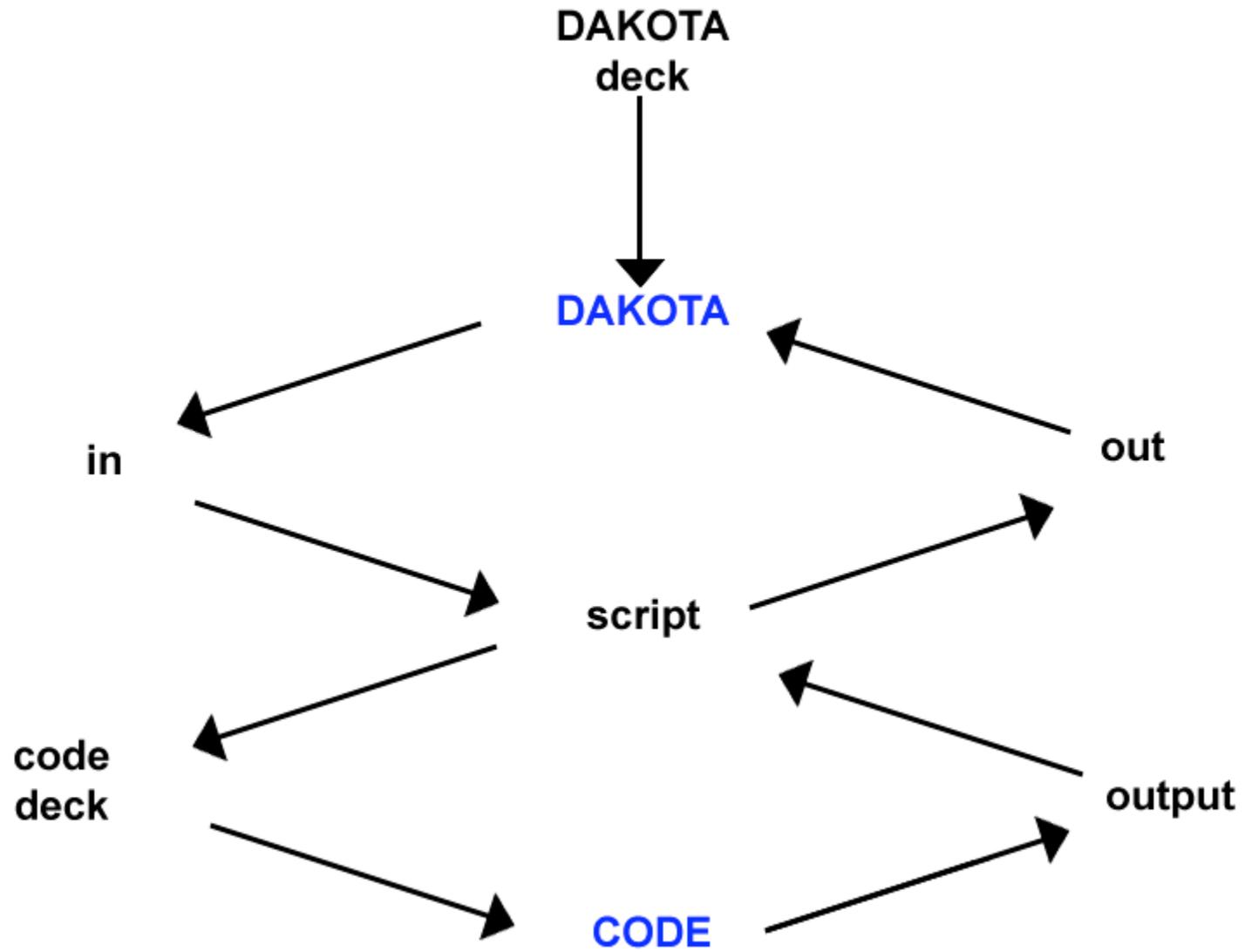
Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.



DAKTOOLS

- Provides tools that fit into the DAKOTA workflow to make runs more powerful and easy
- Provides Code Independence
- Provides Machine Independence
- Maximizes user's efficiency at studying code behavior
- Allows easy structure for users to use and make tools to analyze data
- Implemented in PYTHON for programmability and ease of use





```
#!/bin/csh -f

set file_in = "$1"
set file_out = "$2"

##### generic initializations
(cat ${file_in};\
echo "                { rhom          = 1.0 }"; \
echo "                { gamam        = 1.0 }");) | \
sed \
  -e "s/^ *{ *//\" \
  -e "s/ *}//\" \
>${file_in}.py

sed \
  -e "s/sed1_ld/${file_in}/" \
<sed1_ld3.deck > ${file_in}.tmp

##### create actual input deck
cat ${file_in}.py ${file_in}.tmp >${file_in}.deck

##### execute
time $HDL/bin/hodag ${file_in}.deck >& ${file_in}.hsp

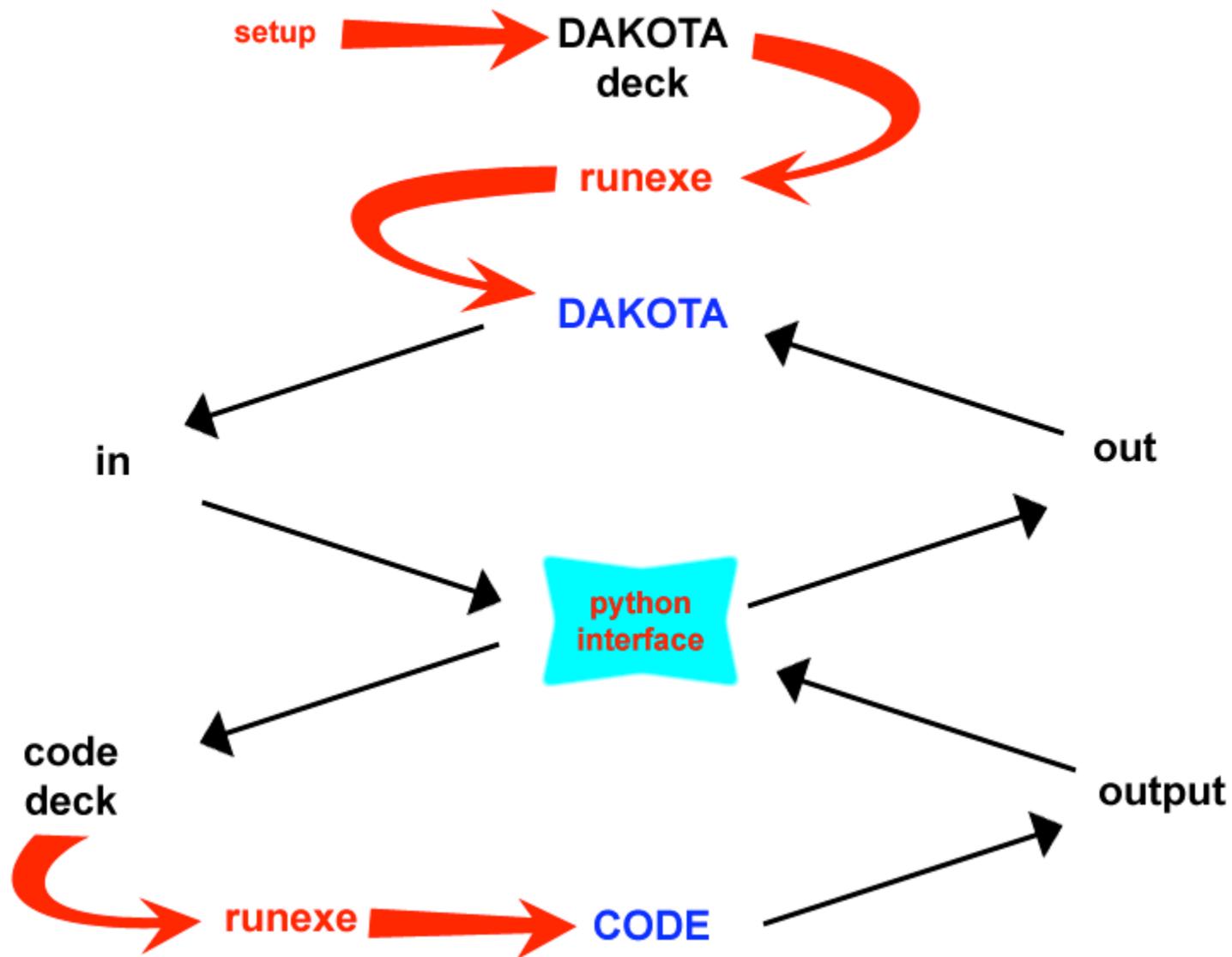
##### Return objective function (shock position at end)

if (-e ${file_in}.su) then
# ..... successfully completed

# Evaluate/return Hodag objective fn (l1 error norm)

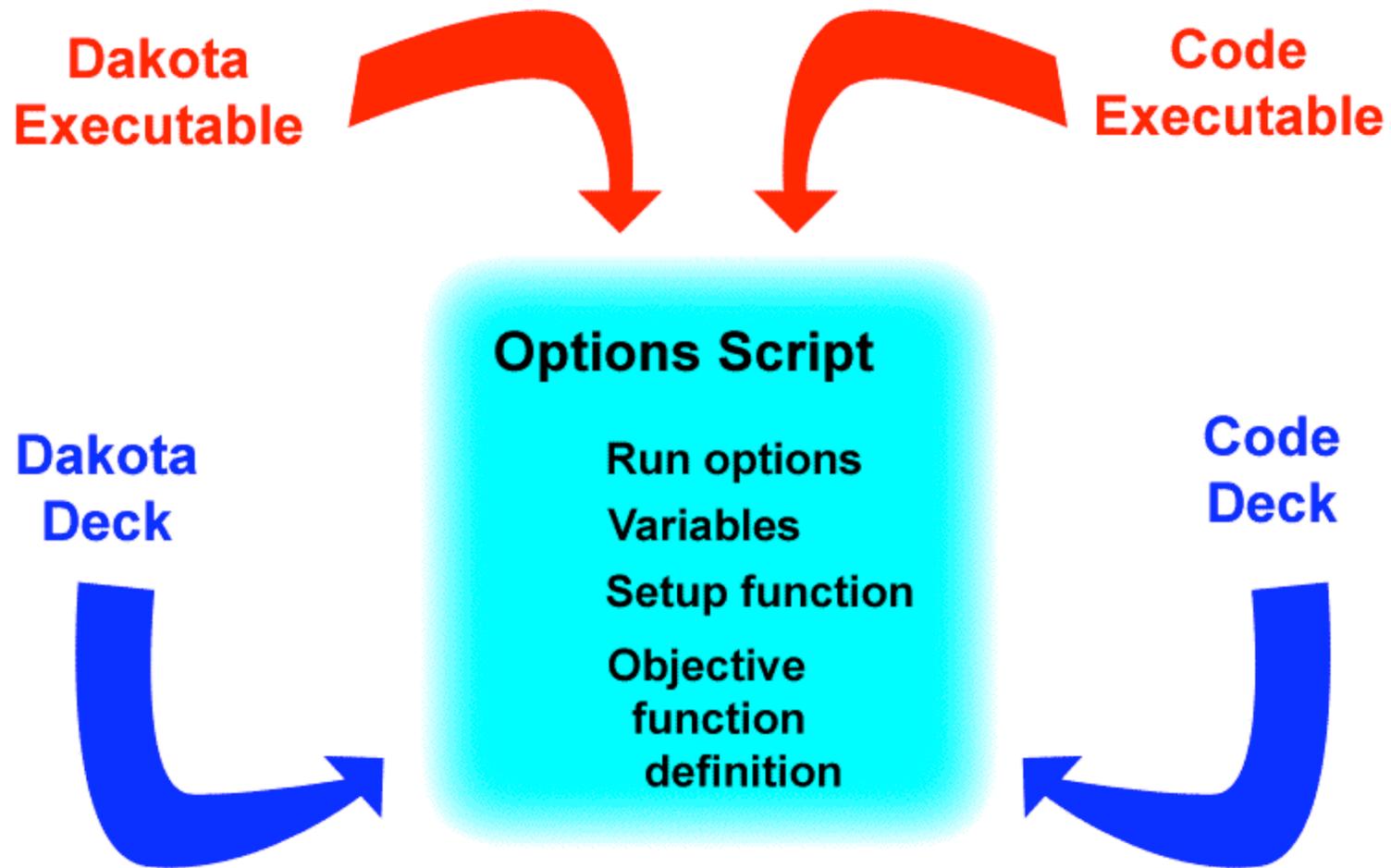
rm -f ${file_in}.tns >&/dev/null
../tabnormu -TSedov ${file_in}.xy >${file_in}.tns
grep "ONE norm=" ${file_in}.tns | head -n 1 | \
  sed \
    -e "s/^.*/          /" \
    -e "s/.*/& f4/" \
  >${file_out}.tmp
cat ${file_in}.su ${file_out}.tmp >${file_out}
else
# ..... ERROR return
echo "                1.0e+99 f0" >${file_out}
endif
```

Simple CSH script used as the interface between
DAKOTA and the simulation code





User Provided Options





User provided options file for daktools script

```
#!/usr/local/bin/python2.3

import os
import sys

### OPTIONS ###
#### these will all have defaults and
#### can be set as command line arguments

systype = os.getenv('SYS_TYPE')

##### global arguments
arguments = {

'problem_name' : 'ar',

##### USER PROVIDED FILES
'dakota_deck'      : 'ar.input',
'code_deck'       : 'ar.gen',

##### DAKOTA
'dakota_path'     : '/g/g14/tipton4/bin/dakota/'+systype+'/dakota',
'dakota_host'    : os.getenv('HOST'),
'dakota_mode'    : 'batch', #interactive | batch
'num_procs'     : '32',

##### CODE
'code'           : 'cretin',
'code_path'     : '/g/g14/tipton4/bin/cretin/'+systype+'/cretin',
'code_host'    : os.getenv('HOST'),
'code_mode'    : 'interactive', #interactive | batch
'code_args'    : '',

##### MISC
'skip_existing_runs' : '1',
'timeout'         : '240' # timeout for batch submission

}

##### Simulation Variables ###
variables = r"""

NE      1e+21  1e+23  1e+24  = log
TE      500    1000   3000   = lin 0 1 2
RMAX    0.01   0.015  0.02

"""

##### setup the run
def setup():
    os.system('ln -s ../argon10.dat argon10.dat')

##### define output function
def output():

    os.unlink('ar.r00')

    numeric = open('ar.plt', 'r')
    numeric.readline()
    numeric.readline()
    numeric.readline()

    analytic = open('../exp.dat', 'r')
    analytic.readline()
    analytic.readline()
    analytic.readline()

    norm = 0.0
    while(1):
        line = analytic.readline()
        if(line == ''):
            break

        avalue = float( line.split()[1] )
        nvalue = float( numeric.readline().split()[1] )

        diff = abs(nvalue - avalue)
        diff = pow(diff,2)

        norm = norm + diff

    return norm
```



Simple Changes to Code Deck are needed

```
c *****  
c alias  
  
alias MODEL argon10.dat  
  
alias ir1 1  
alias ir2 5  
alias rmin 0.0  
alias rmax %%RMAX%% ! 200 micron radius  
  
alias TE %%TE%%  
alias TI TE  
alias NE %%NE%%  
alias NI 0.003125 * NE
```

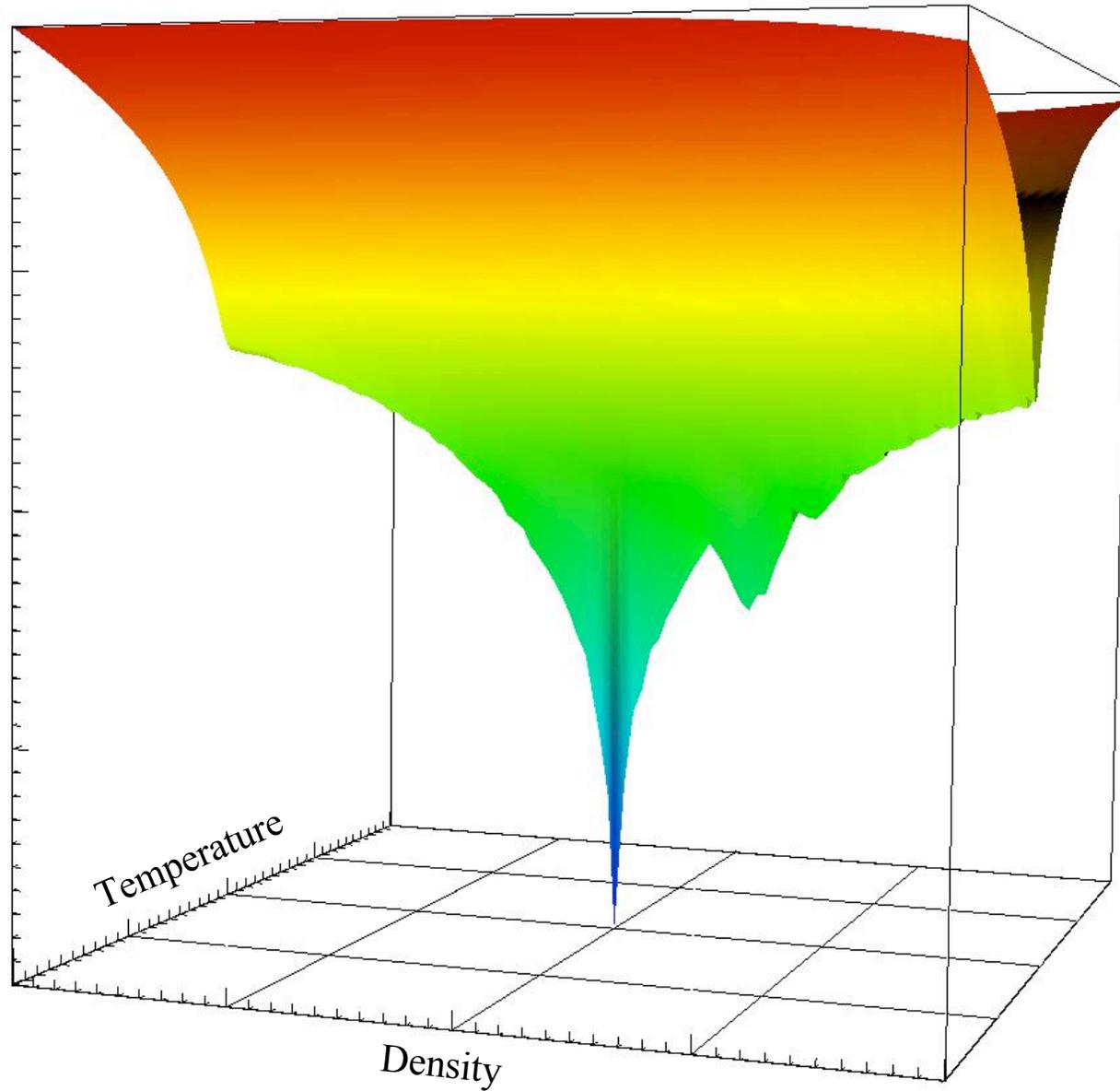


Dictionaries of machine attributes needed for machine independence

```
hostname = ['mcr', 'pengra', 'pvc', 'ilx', 'gps', 'frost', 'blue']
```

```
system_type = {  
  'mcr': 'chaos_2_ia32_elan3',  
  'pengra': 'chaos_2_ia32_elan3',  
  'pvc': 'chaos_2_ia32_elan3',  
  'ilx': 'chaos_2_ia32',  
  'gps': 'tru64_5',  
  'frost': 'aix_5_ll',  
  'blue': 'aix_5'  
}  
  
parallel_cmd = {  
  'mcr': 'srun -N %(nodes)d -n %(procs)d -p pbatch %(cmd)s',  
  'pengra': 'srun -N %(nodes)d -n %(procs)d -p pbatch %(cmd)s',  
  'pvc': 'srun -N %(nodes)d -n %(procs)d -p pbatch %(cmd)s',  
  'ilx': '',  
  'gps': '',  
  'frost': 'poe %(cmd)s -nodes %(nodes)d -procs %(procs)d -rmpool 1',  
  'blue': 'poe %(cmd)s -nodes %(nodes)d -procs %(procs)d -rmpool 1'  
}
```

```
serial_cmd = {  
  'mcr': '',  
  'pengra': '',  
  'pvc': '',  
  'ilx': '',  
  'gps': '',  
  'frost': 'nopoe',  
  'blue': 'nopoe'  
}  
  
cpu_per_node = {  
  'mcr': 2,  
  'pengra': 2,  
  'pvc': 2,  
  'ilx': 1,  
  'gps': 1,  
  'frost': 16,  
  'blue': 4  
}
```



Parameter study on totalb code

Plots the error of each run

The correct answer is the
minimum



DAKTOOLS

- Provides tools that fit into the DAKOTA workflow to make runs more powerful and easy
- Provides Code Independence
- Provides Machine Independence
- Maximizes user's efficiency at studying code behavior
- Allows easy structure for users to use and make tools to analyze data
- Implemented in PYTHON for programmability and ease of use